

Instruction Cycle, Machine cycle and T-state

Time required to complete the execution of an instruction is called Instruction Cycle.
The operations of instruction cycle are given below.

IF (Instruction Fetch)

ID (Instruction Decoding)

OF (Operand Fetch)

EX (Execution)

WB / ST (Write Back / Store)

- **Instruction Fetch:** Fetch the instruction from memory to IR (Instruction Register).
- **Instruction Decoding:** Instruction is decoded and tells the processor what has to be done?
- **Operand fetch:** Fetch the operand from memory if it is required.
- **Execution:** Actual operations (processor operations) are performed.
- **Write Back:** The result will be stored back to the memory location if it is required.

Machine Cycle : Time required to the memory access (memory read or memory write) or IO access is called Machine Cycle.

T state: Time required to complete one sub operation is called T state.

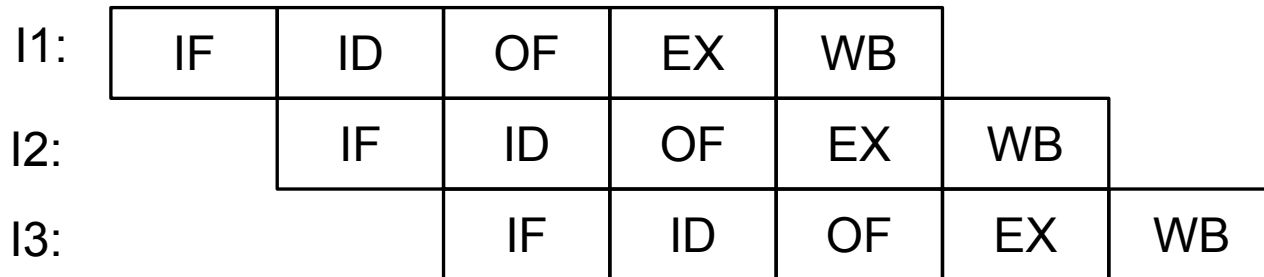
Pipelining

Pipelining is a process of arrangement of hardware elements of the CPU such that its overall performance is increased.

In instruction pipelining, different stages of instructions are overlapped during execution.

There are five stages in instruction pipeline.

IF, ID, OF(optional), EX, WB(optional).

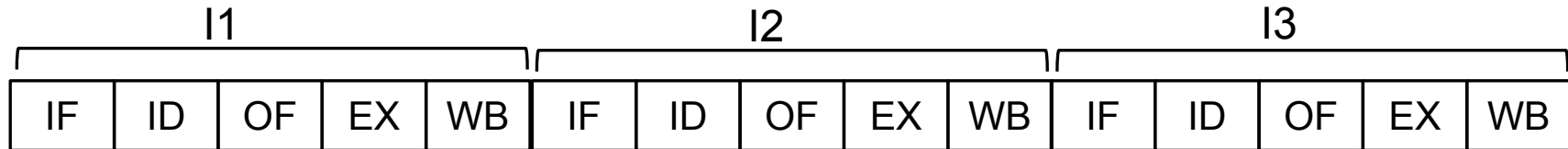


Performance of pipeline processor

Example: A processor has n instructions and k pipeline stages. Each stage requires τ time to complete.

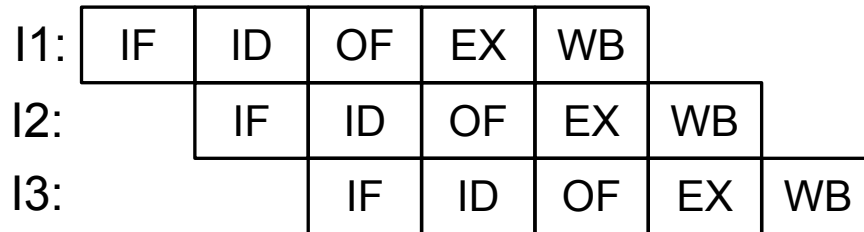
Assume there are $n=3$ instructions and $k=5$ pipeline stages.

Non-pipeline execution



Time required for non-pipeline execution = $3 \times 5 \times \tau = 15\tau$

Pipeline execution

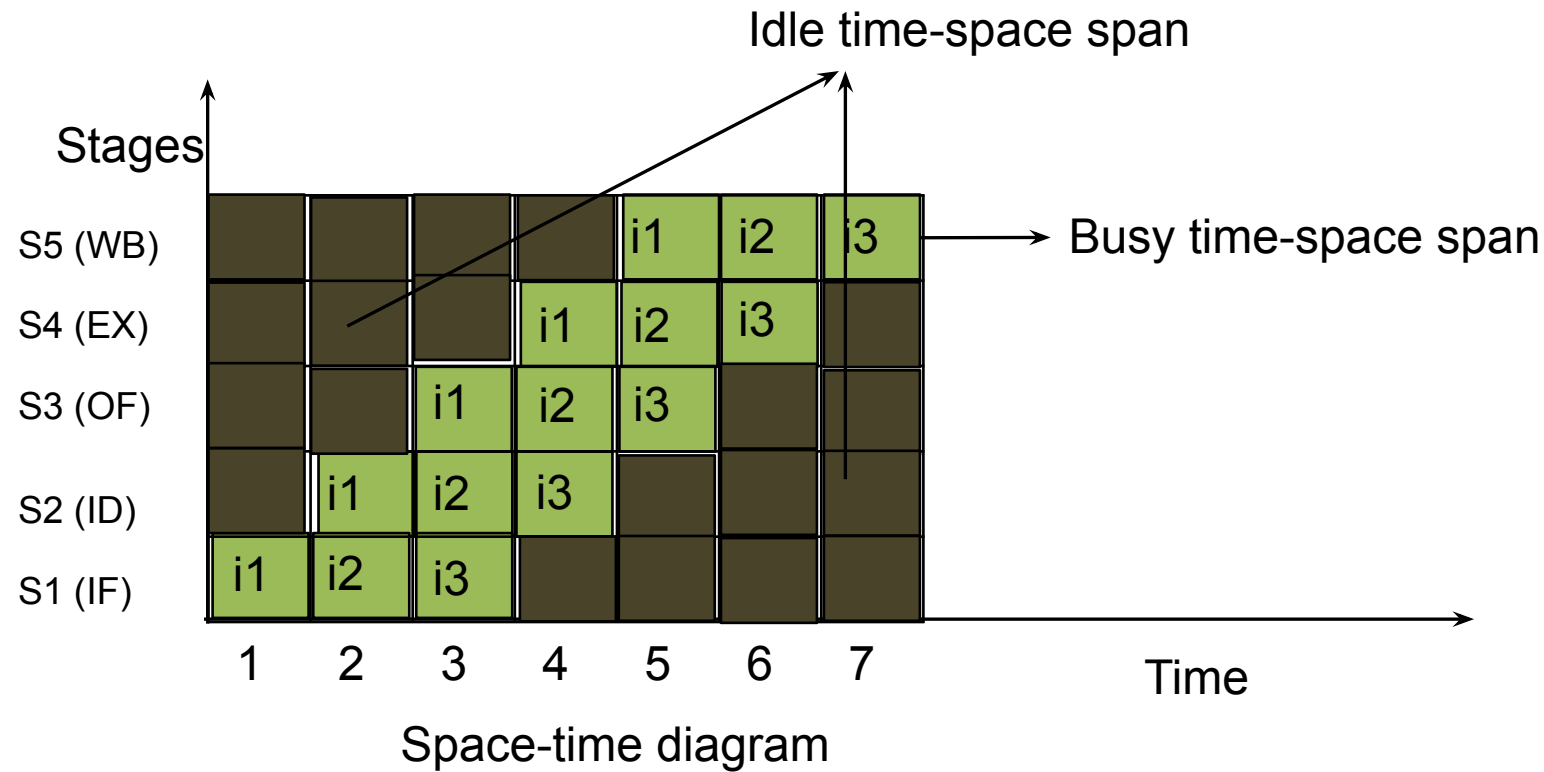


Time required for pipeline execution = $[5 + (3 - 1)] \times \tau = 7\tau$

$$\text{Speed up (S)} = \frac{\text{Time to execute } n \text{ task in } k \text{ stage non-pipeline processor}}{\text{Time to execute } n \text{ task in } k \text{ stage pipeline processor}}$$

$$= \frac{n \cdot k \cdot \tau}{[k + (n - 1)] \cdot \tau} = \frac{n \cdot k}{k + n - 1}$$

Performance of pipeline processor contd.



$$\text{Efficiency } (\eta) = \frac{\text{Busy time-space span}}{\text{Total time-space span}} = \frac{n \cdot k \cdot \tau}{k \cdot [k + (n - 1)] \cdot \tau} = \frac{n}{k + n - 1}$$

Performance of pipeline processor contd.

Throughput: per unit time how many tasks are completed.

$[k + (n - 1)].\tau$ time is required to complete n task

in unit time is required to complete $\frac{n}{[k + (n - 1)].\tau}$ task

$$\text{Throughput } (w) = \frac{n}{[k + (n - 1)].\tau}$$

Pipeline Hazard

In general, there are three major difficulties that cause the instruction pipeline to deviate from its normal operation.

Data hazard arise when an instruction depends on the result of a previous instruction, but this result is not yet available.

Control hazard arise from branch instruction that change the value of PC(Program Counter)

Structural hazard caused by access to memory by two segments at the same time.

i) Data hazard:

Consider the execution of the following instructions:

i1: MUL X, r4, r5 $M[X] \leftarrow r4 * r5$

i2: ADD r2, X, r3 $r2 \leftarrow M[X] + r3$

Here, instruction i2 uses r1 source operand. As a result, i2 can't be correctly executed until r1 is updated by i1.

The solution of this hazard is inserting proper delay between i1 and i2 execution. Incase of inserting delay means just uses no-op instruction which causes no operation but due to this instruction some delay is introduced in the code execution.

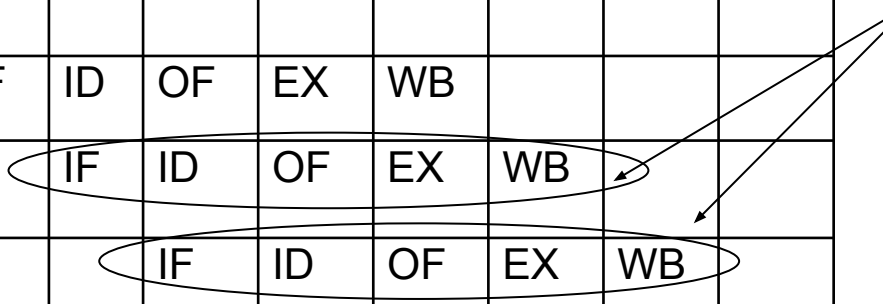
Data Hazard

Clock cycles	1	2	3	4	5	6
1. Instruction1	IF	ID	OF	EX	WB	
2. Instruction2		IF	ID	OF	EX	WB

Pipelining timing with data Hazard

Clock cycles	1	2	3	4	5	6	7	8
1. Instruction1	IF	ID	OF	EX	WB			
2. No-operation		IF	ID	OF	EX	WB		
2. No-operation			IF	ID	OF	EX	WB	
4. Instruction2				IF	ID	OF	EX	WB

Delay



Pipelining timing with inserting delay by no-op instruction

Pipeline Hazard contd.

ii) Control Hazard:

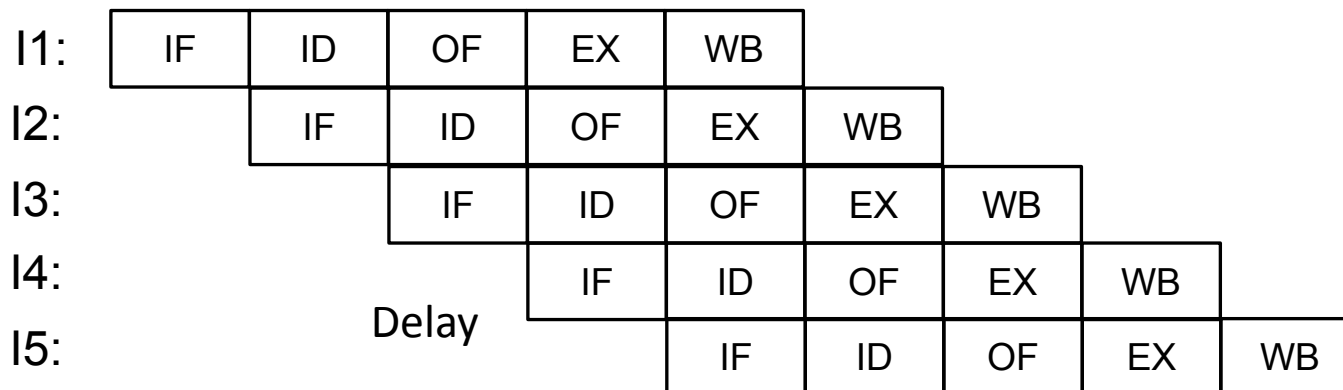
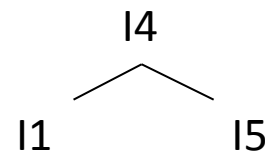
One of the major problems in operating an instruction pipeline is the occurrence of branch instructions.

In a conditional branch, the control selects the target instruction, if the condition is satisfied or the next sequential instruction if the condition is not satisfied.

They can often be solved by delayed branching or branch prediction.

```
do
{
I1:----
I2:----
I3:----
I4 } while(condition);
I5:----
```

I4 is the branching instruction



Pipeline Hazard contd.

iii) Structural hazard:

A structural hazard occurs when a part of the processor's hardware is needed by two or more instructions at the same time.

I1:	IF	ID	OF	EX	WB		
I2:		IF	ID	OF	EX	WB	
I3:			IF	ID	OF	EX	WB

A canonical example is a single memory unit that is accessed both in the fetch stage where an instruction is retrieved from memory, and the memory stage where data is written and/or read from memory.

They can often be resolved by separating the component into different memory (Data memory and program memory).

Thank You